Date=30/10/2020
Lecture By=Manish Mahant
Subject ⇒Package Manager(Node.js)

IN PREVIOUS LECTURE **(QUICK RECAP) Date-29/10/2020**	**In Today's Lecture (Overview)**

JSON - Introduction

JSON: JavaScript Object Notation.

JSON is a syntax for storing and exchanging data.

JSON is text, written with JavaScript object notation.

Exchanging Data

When exchanging data between a browser and a server, the data can only be text.

JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.

We can also convert any JSON received from the server into JavaScript objects.

This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

Package Management Basics

A dependency in your project

A **dependency** is a third-party bit of software that was probably written by someone else and ideally solves a single problem for you. A web project can have any number of dependencies, ranging from none to many, and your dependencies might include sub-dependencies that you didn't explicitly install — your dependencies may have their own dependencies.

A project dependency can be an entire JavaScript library or framework — such as React or Vue — or a very small utility like our human-readable date library, or it can be a command line tool such as Prettier or eslint, which we talked about in previous articles.

What exactly is a package manager?

The package manager will provide a method to install new dependencies (also referred to as "packages"), manage where packages are stored on your file system, and offer capabilities for you to publish your own packages.

In theory you may not need a package manager and you could manually download and store your project dependencies, but a package manager will seamlessly handle installing and uninstalling packages. If you didn't use one, you'd have to manually handle:

- Finding all the correct package JavaScript files.
- Checking them to make sure they don't have any known vulnerabilities.
- Downloading them and putting them in the correct locations in your project.
- Writing the code to include the package(s) in your application (this tends to be done using JavaScript modules, another subject that is worth reading up on and understanding).
- Doing the same thing for all of the packages' sub-dependencies, of which there could be tens, or hundreds.
- Removing all the files again if you want to remove the packages.

Using the package ecosystem

Let's run through an example to get you started with using a package manager and registry to install a command line utility.

First of all, create a new directory to store our experimental app in, somewhere sensible that you'll find again. We'll call it parcel-experiment, but you can call it whatever you like:

```
mkdir parcel-experiment cd parcel-experiment
```

Type the following command, making sure you are inside the `parcel-experiment` directory:

```
npm init
```

You will now be asked some questions; npm will then create a default `package.json` file based on the answers:

- `name`: A name to identify the app. Just press **`Return`** to accept the default `parcel-experiment`.
- `version`: The starting version number for the app: Again, Just press **`Return`** to accept the default `1.0.0`.
- `description`: A quick description of the app's purpose. Type in something really simple, like "A simple npm package to learn about using npm", then press **`Return`**.
- `entry point`: This will be the top-level JavaScript file of the app. The default `index.js` is fine for now — press **`Return`**.
- `test command`, `git repository`, and `keywords`: press **`Return`** to leave each of these blank for now.
- `author`: The author of the project. Type your own name, and press **`Return`**.
- `license`: The license to publish the package under: Press **`Return`** to accept the default for now.

Press **`Return`** one more time to accept these settings.

Go into your `parcel-experiment` directory and you should now find you've got a package.json file. Open it up and it should look something like this:

```
{ "name": "parcel-experiment",

"version": "1.0.0",

"description": "A simple npm package to learn about using npm",
"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "Chris Mills",
"license": "ISC" }
```

Installing parcel

```
npm install parcel-bundler
```

Once that's done *All The Things*, we're now ready for some "modern client-side development" (which really means using build tools to make the developer experience a little easier). First of all however, take another look at your package.json file. You'll see that npm has added a new field, dependencies:

```
"dependencies": {

"parcel-bundler": "^1.12.4"

}
```

Click Below to know more about it
https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Understanding_client-side_tools/Package_management

Questions For self Practice

https://au-assignment.s3.ap-south-1.amazonaws.com/Week_18_Day_5_Challenge-c37497da-836f-4c7b-8cee-97d9d48723cd.pdf